150 ptas.



CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

10021
9 788485 822836

Editorial Delta, S.A.

mi computer CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen II - Fascículo 21

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-90-0 (tomo 2)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 068406
Impreso en España - Printed in Spain - Junio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

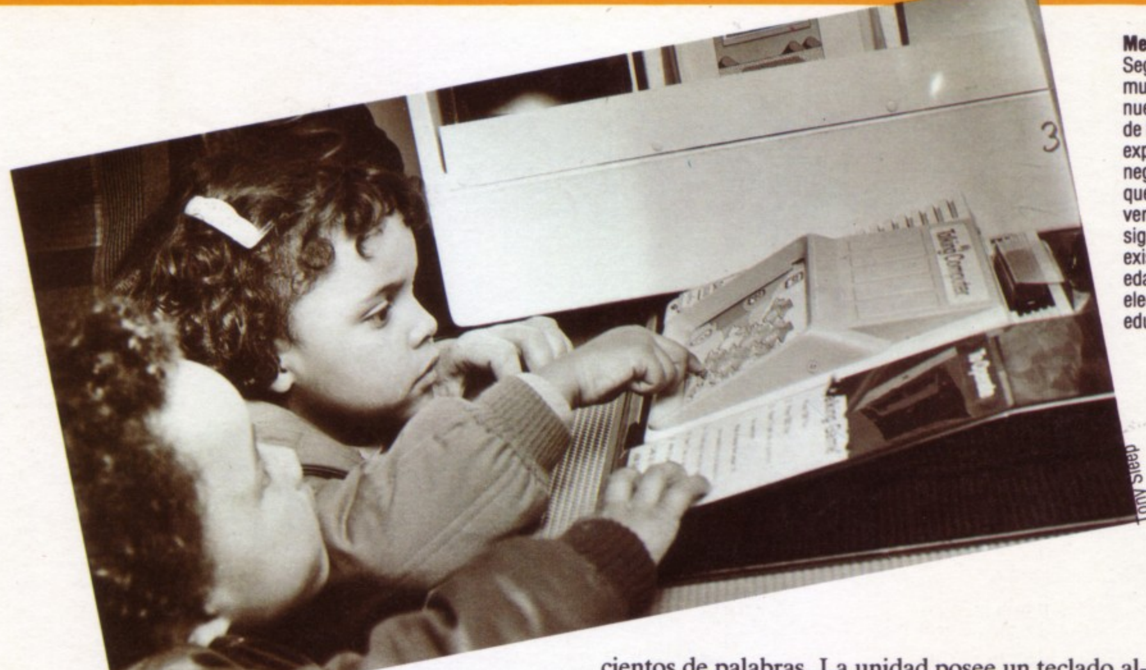
Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.




Tony Sleep

Mentes jóvenes
Según parece, los niños son mucho más receptivos a la nueva tecnología que la mayoría de los adultos, quienes experimentan una reacción negativa ante la idea de tener que aprender ideas nuevas. La versatilidad del microprocesador significa que virtualmente no existe ningún límite mínimo de edad para los juguetes electrónicos y los dispositivos educativos

Para niños

Los más recientes juguetes educativos poseen un potencial de procesamiento equiparable al de un ordenador personal y utilizan técnicas de programación similares

Además de constituir el corazón de todo microordenador, el microprocesador se ha convertido en una configuración estándar de muchos aparatos domésticos, como máquinas de coser, lavadoras e incluso cerraduras para puertas. Los fabricantes de juguetes, asimismo, han hecho experimentos con microprocesadores, especialmente para el control de prototipos de coches y de trenes. Sin embargo, al menos una firma de ordenadores (Texas Instruments) ha descubierto que la producción de juguetes didácticos basados en microprocesadores es muy rentable. La primera incursión de TI en este mercado fue una unidad parecida a una calculadora que planteaba problemas de aritmética elemental. *Little professor* (Pequeño profesor) consiguió una gran popularidad y, aunque haya sido sustituido por *Speak and maths* (Lenguaje y matemáticas), todavía se vende en cantidades significativas.

Speak and maths fue el segundo juguete didáctico de Texas Instruments que empleó el chip TI para síntesis de voz. Éste también se utilizó en el ordenador personal TI99/4A, que se retiró del mercado a fines de 1983, cuando Texas Instruments decidió abandonar la informática doméstica de bajo coste. *Speak and spell* (Hable y deletree), que se lanzó en 1978, posee un vocabulario de algunos cientos de palabras. La unidad posee un teclado alfabético completo (con algunas teclas adicionales) compuesto a partir de una membrana de capas múltiples similar a la del ZX81 de Sinclair. Al pulsar la tecla ENQUIRY (pregunta) al usuario se le solicita de forma audible que deletree una palabra. Cada tecla que se digita se visualiza mediante diodos emisores de luz hasta que se completa la palabra. *Speak and spell* le dice entonces de forma audible al usuario si la ha deletreado de la manera correcta o no.

Speak and maths funciona de modo parecido, pero plantea problemas de aritmética.

Un tercer juguete parlante de TI, *Touch and tell* (Toque y diga), tal vez sea más recreativo que educativo. Utiliza una serie de capas plásticas, cada una de las cuales lleva impreso un patrón o imagen y se identifica por una codificación magnética. Al tocar el niño una zona de la imagen, *Touch and tell* identifica de forma audible el objeto seleccionado.

Mientras que la síntesis de voz es con mucho la técnica de informática más sofisticada que utilizan los fabricantes de juegos y de juguetes, la aplicación más popular son las versiones reducidas de algunos de los juegos recreativos más populares. Deben de existir tantas variedades de este tipo de juegos como juegos recreativos propiamente dichos. Otra área en la que el microordenador ha causado impacto en el mercado del juguete es la de los coches y camiones autodirigidos. Quizá el más conocido de éstos sea el *Big Trak* (véase p. 36), que se programa dando entrada a las instrucciones en un teclado montado en su superficie superior. El juguete se parece a una "tortuga" (véase p. 34), y se puede controlar desde un microordenador a través de su puerta en paralelo.

Otros juguetes basados en microprocesador son: el *Simon*, que invita al niño a repetir una frase musical al azar con luces intermitentes; *Playskool's maximus*, un "maestro" de aritmética similar al *Little professor*, y diversos robots. Entre los diseñados para niños mayores (y para adultos) podemos incluir: *Electroni-kit, Mykit systems* y *Radionics*, que, como sus nombres sugieren, son juguetes electrónicos desmontables que utilizan componentes encerrados en cápsulas, que, una vez armados, se pueden enchufar en un tablero base para formar diversos dispositivos sencillos.



Electroni-kit
Tal como su nombre indica, *Electroni-kit* es un juego para armar que se utiliza para crear dispositivos electrónicos. Sus componentes vienen encerrados en cápsulas de plástico transparente y se enchufan en un tablero base (siguiendo unos diagramas esquematizados que vienen con el juego) para construir el dispositivo deseado. El modelo más sofisticado de la gama incluye los componentes de un microordenador elemental, diseñado para enseñar operaciones muy sencillas en código de lenguaje máquina. Pero con sólo 96 bytes de memoria, no se le puede considerar un ordenador personal

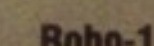

Ian McKinnell
Cortesía de Personal Computer World



Maximus
Éste es un producto muy avanzado de MB Electronics. Aunque se parece a una calculadora, en realidad se trata de otro juego de "emparejamiento", como el *Simon*. El *Maximus*, sin embargo, posee capacidad para operar en diversas modalidades, lo que permite el emparejamiento de notas musicales, imágenes, ritmo, deletreo y formas

Simon
El *Simon* de MB Electronics es una versión en microprocesador de un juego infantil al aire libre. La unidad genera una frase de notas musicales, cada una de ellas acompañada de una luz intermitente. Los cuatro cuadrantes de colores de superficie actúan como interruptores tanto para las luces como para los tonos. El objeto del juego es reproducir exactamente la frase musical

Texas Instruments
TI se introdujo en el mercado educativo a mediados de los años setenta con el *Little professor*, un dispositivo parecido a una calculadora que planteaba problemas de aritmética. Antes de que finalizara la década, Texas Instruments había empezado a comercializar un tutor para deletrear que utilizaba el chip TI para síntesis de voz. Pulsando una tecla, *Speak and spell* (Hable y deletree) solicita que se deletree una palabra. Más recientemente, estas técnicas se han aplicado a juegos de aritmética simple y a dispositivos muy elementales que narran cuentos a los niños más pequeños



Robo-1

El *Robo-1*, de la firma Tomy, es un brazo-robot de diseño convencional que se controla mediante dos palancas de mando. Es incapaz de operar bajo control por programa. Lo más sorprendente del *Robo-1* es su precio: menos del 10 % del precio del brazo-robot didáctico más barato (véase p. 314). Por supuesto, la construcción es mucho menos sólida, ya que no es de metal laminado sino de plástico. El brazo cuenta con la alimentación y el control visuales del usuario, en vez de utilizar motores de precisión a impulsos. Con algo de habilidad, el *Robo-1* se puede conectar a un ordenador personal

Robo-1, cortesía de Hamleys

Big Trak

Aunque por su aspecto se asemeja a algunos de esos vehículos de juguete resistentes para los niños más pequeños, el *Big Trak* es en realidad un robot móvil encubierto. Totalmente autocontenido, se programa dando entrada a los códigos de dirección y distancia en un teclado dispuesto sobre su superficie. Con un poco de habilidad se puede conectar el *Big Trak* a un microordenador personal por medio de una puerta en serie o en paralelo. El vehículo se podría guiar entonces bajo control de programa, lo cual abriría la posibilidad de bifurcar a un subprograma diferente en caso de que hubiera que hacer frente a determinadas condiciones

Cortesía de Milton Bradley Ltd

Ian McKinnell

Verificadores de ortografía

Muchos procesadores de textos disponen de programas verificadores de ortografía, y también están empezando a aparecer otros que se encargan del estilo y de la gramática

Aún está muy lejano el día en que los diseñadores de ordenadores puedan crear máquinas que posean capacidad de generar y manipular idiomas humanos. Una de las aplicaciones proyectadas para la quinta generación de ordenadores, que debería aparecer en el transcurso de la década de 1990, es la traducción mecánica entre, por ejemplo, el inglés y el japonés. Ya hay facilidades para la traducción mecánica de una prosa relativamente sencilla, como la de los expedientes e informes oficiales, si bien los borradores que producen los ordenadores de unidad principal invariablemente se han de corregir y pulir a mano. Las anécdotas sobre errores abundan: se cuenta que el dicho "El espíritu está pronto, pero la carne es débil", se tradujo del inglés al ruso y nuevamente al inglés mediante dos programas diferentes. El resultado final fue: ¡"El vino es agradable, pero la carne se echó a perder"!

Las anécdotas apócrifas de este tipo sirven para ilustrar un punto importante: las dificultades que surgen cuando un ordenador está procesando datos sin entender lo que éstos significan. A los estudiantes de informática se les suele plantear el problema de considerar cómo podría distinguir un ordenador los significados de estas oraciones en inglés:

TIME FLIES LIKE AN ARROW
(El tiempo vuela como una saeta)

FRUIT FLIES LIKE A BANANA
(A las moscas de la fruta les gusta el plátano)

La construcción de ambas oraciones es idéntica, pero en el primer caso FLIES es un verbo ("vuela"), mientras que en el segundo forma parte de un predicado nominal ("moscas"). Nosotros podemos reconocerlas como distintas a través de la experiencia. En un ordenador, la experiencia se puede simular, disponiendo de la suficiente memoria, pero esto cae en el campo de la inteligencia artificial, y la investigación en esta área no está muy avanzada. En realidad estamos aludiendo a la diferencia entre "sintaxis" y "semántica". La sintaxis, que son las reglas relativas a los procesos de construcción que se utilizan en un idioma, es algo que los ordenadores pueden manejar con bastante facilidad, como bien lo saben todos los programadores de ordenadores que se hayan encontrado con mensajes de SYNTAX ERROR? (¿error de sintaxis?). La semántica, en cambio, se refiere al significado que comunican aquellas frases y construcciones.

En los años cincuenta, Noam Chomsky desarrolló las bases de la teoría actual sobre los lenguajes humanos y las reglas de la gramática, y aunque no tuvo una relación directa con las ciencias de la informática, sus teorías son perfectamente aplicables tanto a la traducción mecánica como a la escritura de intérpretes y compiladores para lenguajes de programación.

Consecuencia directa de las investigaciones del lingüista norteamericano ha sido la creación de diversas herramientas de software para ayudar a la escritura de texto. Además de los paquetes para tratamiento de textos, auxiliares de la creación, la edición y la impresión de texto, hay programas para la corrección de documentos, para la detección de errores de ortografía y mecanografía, e incluso para verificar la gramática y el estilo de la escritura.

Todos los programas verificadores de ortografía se valen de un diccionario retenido en disco, que suele almacenar entre 25 000 y 50 000 palabras. La mayoría de los paquetes permiten que el usuario incorpore al diccionario nuevas entradas.

Constituye un problema, sin embargo, hallar el espacio de memoria adecuado para un diccionario completo. Se sabe que un byte de ocho bits puede retener un solo carácter alfanumérico empleando el código ASCII. Por consiguiente, aun concediendo un promedio muy optimista de apenas cinco caracteres por palabra, un diccionario de 30 000 entradas requeriría 150 Kbytes de almacenamiento, que es mucho más de lo que permiten la mayoría de las unidades de disco para ordenadores personales. Por suerte, se puede comprimir este tipo de datos mediante la utilización de dos técnicas.

La primera da por supuesto que el diccionario sólo incluirá letras en minúscula (una rutina del programa de ortografía manipulará las conversiones), y que no se necesitarán dígitos numéricos ni ciertos signos de puntuación, por lo que todos éstos se pueden eliminar del programa. En consecuencia, podríamos construir nuestro diccionario utilizando un máximo de 32 caracteres diferentes, en lugar de la gama ASCII completa de 128 (o 256, incluyendo los símbolos para gráficos). Por lo tanto, podríamos reducir las necesidades de almacenamiento de cada carácter de ocho bits a cinco. La palabra "computer", por ejemplo, se podría almacenar en un total de 40 bits, o cinco bytes. Los cinco primeros bits del primer byte especificarían la letra "c", y los tres bits siguientes, más los dos primeros del segundo byte, especificarían "o", y así sucesivamente.

La segunda técnica que se emplea en los verificadores de ortografía parte de la premisa de que ciertas combinaciones de caracteres aparecen con tanta frecuencia que se podrían representar en un único byte. Éste se señalaría de alguna forma mediante una "bandera" para indicar que se trata de un distintivo común para un grupo de caracteres y no de un solo carácter. Casi con toda seguridad su orde-

nador personal utiliza esta técnica en BASIC: cada palabra clave, como PRINTo NEXT, está almacenada en RAM en un solo byte para ahorrar espacio.

En un diccionario verificador de ortografía, este segundo sistema se emplea al comienzo de las palabras. Consideremos, por ejemplo, el gran conjunto de palabras que incluyen prefijos como "re", "in", "des" o "auto". El paquete VizaSpell, que se ejecuta con el procesador de textos VizaWrite en el Commodore 64, utiliza ambas técnicas para introducir un diccionario de 30 000 palabras en sólo 65 Kbytes de disco.

El trabajo que más dificultades entraña a un verificador de ortografía es, sin embargo, el de buscar en el diccionario todas las palabras que componen el documento. Se podría emplear una búsqueda binaria (véase p. 416), pero, tratándose de un documento de mil palabras, esto ocuparía horas. Lo ideal sería que el procesador de textos verificara cada palabra a medida que ésta se digitara, pero esto no es práctico en términos de programación y, en consecuencia, un documento por lo general se deberá verificar como un todo, ya sea en disco o (como en las máquinas mayores) en RAM. El programa funciona a lo largo del documento y compila en orden alfabético una lista de las palabras que contiene. A menudo, más del 50 % de un extenso informe consta de sólo 100 palabras diferentes.

```
Vizaspell                  name:hamlet
  3 a                   2 of
  1 against             1 opposing
  4 and                 2 or
  1 arms                1 outragous
  1 arrows              1 qusetion
  2 be                  1 say
  2 by                  1 sea
  1 conssumation        1 shocks
  1 die                 1 sleeep
  2 end                 1 sleep
  1 flesh               1 slings
  1 fortune             1 suffer
  1 heartake            1 take
  1 hier                2 that
  1 in                  5 the
  2 is                  1 them
  1 mind                1 thoudsand
  1 more                2 tis
  1 natural             8 to
  1 no                  1 truobles
  1 nobler              1 we
  1 not                 1 wether
     Spellcheck complete - press space
```

La mayoría de los verificadores de ortografía emplean este sistema para proporcionar un útil informe adicional acerca del empleo de palabras en su documento, lo que puede resultar de gran ayuda para detectar repeticiones innecesarias. Después, un algoritmo sencillo repasa esta lista y la lista del diccionario simultáneamente, buscando los emparejamientos. De este modo, el tiempo que lleva completar la búsqueda se reduce mucho y es constante: cuatro minutos en el caso del VizaSpell, independientemente de la longitud del documento.

Las palabras que no se encuentran en el diccionario se imprimen en forma de lista o bien se señalan en el mismo documento. Ante cada palabra señalada al usuario se le presentan tres opciones:

1) En la palabra hay un error de ortografía o de mecanografía que se debe corregir;
2) La palabra es correcta y debería agregarse al diccionario del programa;
3) La palabra es correcta, pero es poco probable que se vuelva a emplear en otra ocasión (p. ej., forma parte de unas señas), de modo que se debe dejar así, sin incorporarla al diccionario.

Los verificadores de gramática y de estilo funcionan de manera similar. Los primeros trabajan de acuerdo con una cantidad limitada de reglas (como cuidar que haya una letra mayúscula al comienzo de cada oración) y, por consiguiente, hay muchos errores gramaticales que no se pueden detectar. Los verificadores de estilo están aún en una etapa incipiente, y la mayoría de los paquetes habituales utilizan tan sólo un gran diccionario de ejemplos con el fin de identificar la sintaxis y las expresiones erróneas.



Popperfoto

```
Vizawrite                  name:hamlet
On Page  1 of  1          Line  1 Col  5
   To be, or not to be:
   That is the qusetion:
   Wether 'tis nobler in the mind
     to suffer
   The slings and arrows of outragous
     fortune
   Or to take arms against a sea
     of truobles
   And by opposing, end them?
     To die, to sleep;
   No more-and by a sleeep to say
     we end
   The heartake and the thoudsand
     natural shocks
   That flesh is hier to, tis a
     conssumation.
   Devoutley to be wished.
X (End of Page)
```

Ian McKinnell

"Ser o no ser"
¡Imagínese cuánto más fácil sería la asignatura de Lengua y Literatura si se pudieran usar programas verificadores de ortografía en el aula de exámenes! Podemos utilizar el monólogo de *Hamlet*, de Shakespeare, para ilustrar cómo funciona uno de los programas de este tipo (el VizaSpell). Primero se digita el texto en el ordenador empleando un procesador de textos. Luego se llama al verificador de ortografía mediante un par de órdenes simples, y éste crea, en orden alfabético, una lista de todas las palabras utilizadas, incluyendo también su frecuencia de uso. Esta lista se coteja con el diccionario en disco, y se destacan los términos que no se reconocen. Cuando se lo emplea por primera vez, es probable que el programa "ilumine" algunas palabras aparentemente comunes, que se pueden incorporar al diccionario para una utilización posterior

Escribir en BASIC un tipo sencillo de verificador de ortografía, de gramática o de estilo puede ser un ejercicio muy interesante incluso para un programador poco experimentado, aunque para ello se necesitará tener un conocimiento bastante profundo de las funciones de la máquina para manipulación de series. Al aumentar la sofisticación del software es de esperar que los paquetes para tratamiento de textos vengan con dichas funciones ya incorporadas. Desde luego, a cualquier escritor le encantaría: 'ORDEN > REDACTAR ARTICULO, LONGITUD 1200 PALABRAS, EMPEZAR'.

Programas sofisticados

Los generadores de aplicaciones son parecidos a los generadores automáticos de programas, pero además de las aplicaciones de gestión poseen también aplicaciones para juegos



IAN McKINNELL

Pinball Construction Set
Este paquete es un tipo de generador de aplicaciones para juegos. El usuario diseña el trazado y la lógica para un juego de "millón" utilizando un menú de objetos y diversas herramientas representadas gráficamente para fijarlas sobre el tablero

En un capítulo anterior de Mi Computer analizamos una clase de programa para ordenador que, tras dar el usuario un conjunto de especificaciones, produce un programa capaz de llevar a cabo la aplicación requerida. Dichos generadores de programas se pueden adquirir para la mayoría de los micros de gestión y existen algunos paquetes para ordenadores personales, aunque el tipo de aplicaciones para los que son idóneos requiere al menos una unidad de disco.

Una forma mucho más común de generar programas para requerimientos específicos implica la utilización de paquetes denominados *generadores de aplicaciones*. A diferencia de los generadores de programas, éstos producen programas que no son autónomos sino que necesitan el paquete generador de aplicaciones original para poder ser ejecutados. Consideremos la creación de un programa para manipular la facturación con utilización de estos dos tipos de generadores, para ilustrar las diferencias entre ambos.

Si fuéramos a utilizar un generador de programas, primero el software se habría de cargar desde el disco al ordenador. Una vez el usuario hubiera respondido a todas las preguntas relativas a archivos, registros, campos, relaciones matemáticas, trazados de pantalla e informes impresos requeridos (es decir, una vez que hubiera especificado el programa de aplicaciones deseado), el generador le pediría que insertara en la unidad de disco un disco vacío. Entonces guardaría el nuevo programa que habría generado en este segundo disco. Este proceso se podría repetir y se podría hacer una copia del programa de facturación para cada sucursal de la empresa.

En comparación, un generador de aplicaciones parece inicialmente menos satisfactorio. Una vez cumplida la etapa de especificaciones, las rutinas necesarias se graban en el mismo disco que el generador. También podría grabar el programa en un disco separado, pero lo haría de forma tal que aún se necesitaría el disco del generador original para poder ejecutar la aplicación. Aunque se podría utilizar una única copia del paquete original para producir un número ilimitado de aplicaciones diferentes, se desprende que todas ellas se deben utilizar en la misma posición física que el disco del generador. Si usted quisiera poner su aplicación a disposición de otros usuarios, éstos necesitarían adquirir asimismo una copia del generador. Por supuesto, los generadores de este tipo emplean diversos métodos para proteger el programa, con el fin de que resulte muy difícil el poder copiarlo sin la debida autorización.

Un generador de aplicaciones es en realidad un sofisticado programa para fines generales. Cuando el usuario especifica su aplicación, sencillamente está asignando valores a una cantidad de variables importantes dentro del generador, denominadas *parámetros*. Éstos controlan el flujo del programa, la estructura de los datos y los trazados para la pantalla y la impresora. Cuando la aplicación se guarda en disco, lo que en realidad se está almacenando es una lista de estas variables o parámetros. Esta lista (a la que en ocasiones se alude como un "módulo de aplicación") actúa, por tanto, como un conjunto de instrucciones que le dicen al generador de aplicaciones cómo tiene que efectuar una aplicación determinada.

Algunos paquetes llegan más lejos y permiten que el usuario especifique su aplicación en forma de un lenguaje de muy alto nivel (similar al seudolenguaje que utilizamos por primera vez al desarrollar una nueva rutina en el curso de programación BASIC). Este listado lo interpretará el generador, y éste, a su vez, podría ser interpretado por el intérprete de BASIC si el programa de generación estuviera escrito en este lenguaje, lo cual constituye un caso especialmente interesante de jerarquía de software (véase p. 66).

No es raro que los módulos de aplicaciones los creen y comercialicen empresas distintas de la de los autores del generador original. Por ejemplo, dBase II (el más popular de todos los paquetes sofisticados de base de datos que existen para microordenadores) en realidad se puede considerar como un generador de aplicaciones, que contiene módulos consistentes en series de órdenes para base de datos de alto nivel. Los módulos para aplicaciones menos corrientes (como un sistema de contabilidad pensado para corredores de bolsa) se pueden construir sin tener que escribir el programa partiendo desde cero. Dadas las limitadas dimensiones del mercado, un paquete para corredores de bolsa que se ejecutara bajo el dBase II bien podría ser mejor que uno escrito en BASIC, porque el autor del programa tendría que haber concentrado todos sus esfuerzos en la *operación* del programa y no en la *escritura* del código. Las partes del programa más susceptibles de error (p. ej., la manipulación de archivos) serán escritas por los autores del generador y serán probadas por miles de usuarios en distintas aplicaciones.

Pero la diferencia principal entre un generador de programas y un generador de aplicaciones radica en su facilidad para el usuario. El programa final creado por un paquete generador de programas consistirá en código escrito artificialmente, por lo general en un lenguaje como el BASIC. Dicho código será inferior, tanto en eficacia como en estilo, al código generado por los humanos. En el caso del generador de aplicaciones, sin embargo, hasta un 99 % del programa final consistirá quizá en código escrito por la casa de software y éste estará, con toda probabilidad, también en código de lenguaje máquina. Éste es el caso del Silicon Office, uno de los generadores de aplicaciones más sofisticados y más fáciles de utilizar que existen para microordenadores de gestión. El programa resultante será más rápido y más eficaz, incorporará procedimientos de revisión para detectar posibles errores del operador y producirá visualizaciones en pantalla activadas por menú y de trazado claro.

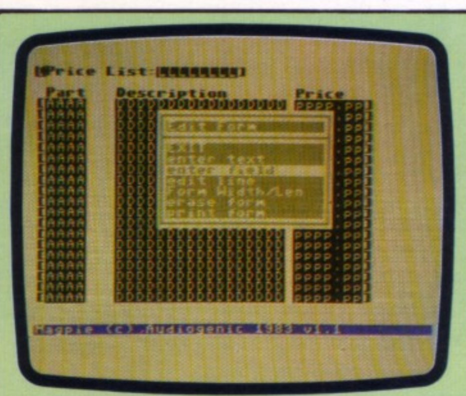
Además, los generadores de aplicaciones no están restringidos a los programas de gestión. Tal vez el mejor ejemplo de un paquete que no sea de gestión es el Pinball Construction Set (véase p. 241), en el cual el módulo de aplicaciones se especifica efectivamente trazando los elementos de la máquina de "millón" (billar electrónico) requerida.

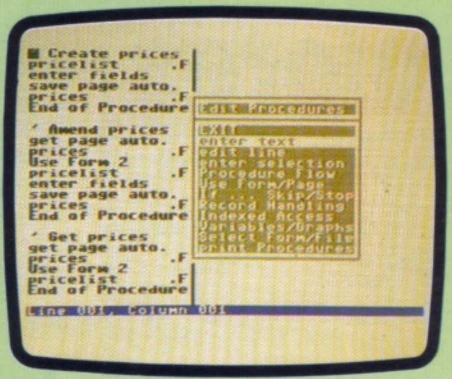
Existen, de hecho, muchos puntos en común entre este tema y la programación orientada de objetos, de la que ya hemos hablado (véase p. 242), pero se pueden resumir en líneas generales en los siguientes términos: estimular al programador para que implemente sus aplicaciones especificando simplemente los objetos que del programa se requieren. Incluso los programas de hoja electrónica sencillos, existentes para ordenadores personales como el Spectrum de Sinclair, se pueden considerar generadores de aplicaciones: el usuario simplemente especifica la relación entre los diversos campos, y el paquete hace por él todo el trabajo rutinario.

Magpie (fabricado por Audiogenic para el Commodore 64 con una unidad de disco) es un generador de aplicaciones preparado para aplicaciones de gestión u otras aplicaciones serias. Se trata de otro paquete que hace buen uso de la programación orientada de objetos visuales: las relaciones entre datos de distintos registros se especifican al diseñar el trazado de esos registros.

A pesar de que no se consideran estrictamente generadores de caracteres, en la actualidad un número creciente de paquetes están incorporando algunos de estos principios. Al ejecutarlos por primera vez, los programas "activados por parámetros" de este tipo formulan al usuario una serie de preguntas y graban las respuestas en disco a lo largo del programa. Esta información determinará algunos de los detalles de la operación del programa. Un programa de facturación, por ejemplo, formularía preguntas relacionadas con la información que a la empresa le interesa incluir en cada factura, y los plazos de crédito estándar que concede. Un juego recreativo preguntaría al usuario con cuántos extraterrestres, bases y cohetes iniciará el juego, o incluso le permitirá diseñar los invasores.

El software se inclina cada vez más a evitar que el usuario tenga que aprender a programar, proporcionando al mismo tiempo un gran nivel de flexibilidad a la operación. Sería muy positivo que el propio software se adaptara a las exigencias del usuario (en vez de que sea el usuario quien deba adaptarse al software).







Cortesía de Soft

El Magpie para el Commodore 64

En la primera etapa de la creación de una aplicación se especifica el trazado de los modelos, transacciones e informes, como esta lista de precios. Llenando las columnas con letras (A, D, P), se especifican los campos de la base de datos a utilizar

A continuación se especifican todos los cálculos y procesos en forma de lista de instrucciones en un lenguaje de programación de alto nivel, que Magpie interpretará. Aquí vemos las rutinas para corregir los precios y para cargarlas (GET) desde el disco

Magpie es activado por menú. A medida que se selecciona una opción (p. ej., CREATE), aparece otra junto a ella mostrando todas las opciones CREATE. Esta pantalla visualiza el resultado de seleccionar CREATE, DISK, DELETE (borrar) y el archivo a borrar, en este caso PRICE LIST (lista de precios)

Ian McKinnell

Imitando instrumentos

La orden ENVELOPE del BBC Modelo B proporciona un control casi ilimitado

En uno de los capítulos anteriores se analizó el formato de la orden SOUND del BBC Micro. No obstante, para examinar con toda profundidad las capacidades de sonido del BBC, es necesario utilizarlo con la versátil orden ENVELOPE. Ésta permite al usuario formar cuatro sonidos, hasta el punto de que se pueden programar imitaciones bastante aceptables de instrumentos convencionales. Además, los efectos sonoros para juegos se pueden refinar para que suenen de manera muy parecida a las explosiones o los disparos.

ENVELOPE se construye de la siguiente manera:

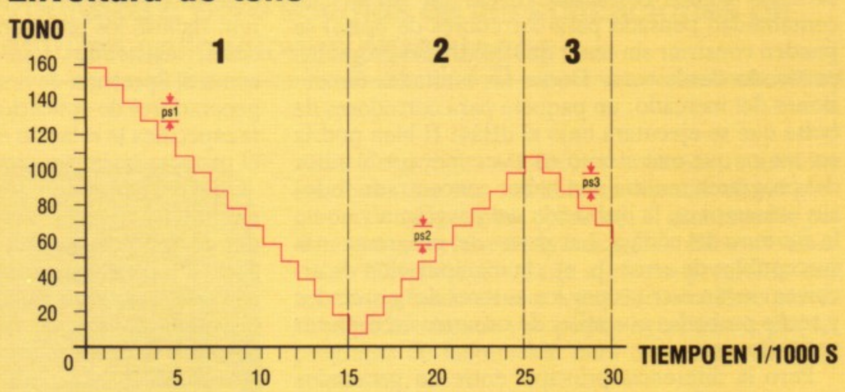
ENVELOPE N,T,PS1,PS2,PS3,NS1,NS2,NS3,
AR,DR,SR,RR,FAL,FDL

El primer parámetro, N, establece el número de envoltura y sirve para identificar ésta con las órdenes relacionadas SOUND o SOUND &. Se puede sustituir una de las cuatro envolturas para el volumen fijado (V) establecido por SOUND mediante un número negativo (de 0 a −15; véase p. 388).

T (de 0 a 127) y (de 128 a 255)

Éste es el control del tiempo para la orden. Establece la duración de cada intervalo de la construcción de la envoltura en centésimas de segundo. Por consiguiente, T = 5 significa que cada intervalo de envoltura dura cinco centésimas de segundo (0,05 segundos). Agregando 128 a la duración del intervalo requerida se suprimirá la repetición automática de la envoltura de tono, de modo que T establecido en 5 + 128 = 133 da una duración de intervalo de cinco centésimas de segundo para una envoltura de tono que se produce una sola vez en la nota. La utilización del término "envoltura de tono" puede parecer algo confusa dado que previamente

Envoltura de tono



la envoltura se ha empleado en términos de volumen, pero en este caso se refiere a la variación del tono en el transcurso de la duración de una nota. Esta configuración posee poco valor desde el punto de vista musical, a menos que se desee un "vibrato", pero puede ser útil para conferir a los efectos sonoros interesantes "trinos". Como se ve en el diagrama, la envoltura de tono se divide en tres secciones. La respuesta de cada sección se puede determinar mediante los números relacionados con PS y NS, del siguiente modo:

De colores

La utilización de sprites en el Commodore 64

Una de las configuraciones más atractivas del Commodore 64 es su capacidad para el empleo de sprites. Los sprites se construyen del mismo modo que los caracteres definidos por el usuario, pero son mucho más grandes, ya que constan de 21 filas de 24 pixels. Los sprites no se visualizan en la matriz normal para caracteres en pantalla y ello permite que se pueda mover un pixel cada vez, en vez de exigir ocho pixels para mover un carácter de una celda a la siguiente. En la pantalla se pueden visualizar hasta ocho sprites a la vez, cada uno con estas características individuales de programación:

Forma y color

Un sprite se define de forma muy similar a un carácter de 8 × 8 pixels, pero se necesitan 63 bytes para retener los patrones codificados en forma binaria. Una vez que se ha definido la forma de esta manera, se obtiene un bloque de 63 posiciones consecutivas. Cada sprite posee un indicador de datos que señala la zona de la que proviene la forma del sprite, lo cual significa que puede haber más de un sprite que "mire" hacia la misma zona de memoria; es decir, que puede haber sprites idénticos. También se puede cambiar su forma haciendo que su indicador mire una zona diferente de memoria.

Cada sprite se puede colorear con alguno de los 16 colores existentes. Asimismo, se puede pintar de varios colores, con el inconveniente de que a veces la resolución horizontal se divide en dos.

Tamaño y movimiento

Los sprites se pueden ampliar horizontal y verticalmente, o en ambas direcciones, para duplicar su tamaño original. El sprite totalmente ampliado mide 48 × 42 pixels. En este caso también se ha de pagar un precio por ello: la resolución se divide por dos en la dirección de la ampliación.

PS1,PS2 y PS3 (de −128 a 128)

PS indica *pitch step* (intervalo de tono). La orden SOUND establece el tono al comienzo de la nota. PS1 establece la modificación, positiva o negativa, de tono por intervalo para la primera sección, PS2 para la segunda sección y PS3 para la tercera sección. PS, al igual que SOUND, se establece en cuartos de semitonos.

NS1,NS2 y NS3 (de 0 a 255)

NS quiere decir *number of steps* (número de intervalos) por sección; y, conjuntamente con PS, selecciona la velocidad a la cual se modifica el tono en una sección y también la duración de toda la envoltura de tono. Los valores PS y NS para el ejemplo anterior son los siguientes:

T = 1	PS1 = −10	NS1 = 15
	PS2 = +10	NS2 = 10
	PS3 = −15	NS3 = 5

En este caso, el tono se establece mediante SOUND = 160. El resultado es:

ENVELOPE 1,1,−10,10,−15,15,10,5,0,0,0,0,0,0

La duración de la envoltura viene dada por (NS1+NS2+NS3) × T, que en este caso es (15+10+5) × 1 = 0,3 segundos. Normalmente, la envoltura de tono se repetirá de manera automática en el transcurso de la duración de una nota, a menos que lo impida el parámetro de tiempo, T.

En el próximo capítulo de *Sonido y luz* volveremos a ocuparnos de las configuraciones de sonido del BBC Micro y explicaremos el funcionamiento de la envoltura de volumen.

Un sprite se puede desplazar un pixel cada vez y la posición antigua se borra de forma automática. Los sprites también se pueden mover para salir del área normal de visualización de la pantalla o para entrar en ella.

Prioridad y colisión

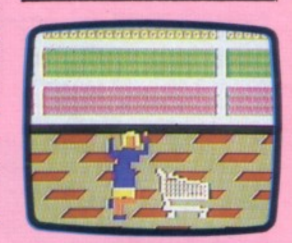
Cuando dos sprites se cruzan en su camino, uno aparece como si pasara por delante del otro. Si hay algún agujero en el sprite que pasa por delante, a través del mismo se verá el sprite que pasa por detrás. Se puede utilizar la prioridad para conseguir algunos interesantes efectos tridimensionales. A cada sprite se le da un número de 0 a 7 y la sencilla ley de la prioridad consiste en que los sprites de números menores pasan por delante de los sprites de números mayores. Por regla general, los sprites se mueven por delante de cualquier carácter normal que haya en la pantalla, pero también se pueden programar de tal manera que pasen por detrás de ellos.

Esta última característica se puede utilizar para dar la impresión de profundidad en la pantalla.

Cuando dos sprites se cruzan, se señala en un registro de colisiones. Mirando (PEEK) en este registro, el programador puede saber cuáles son los sprites implicados. Existe otro registro similar que señala cuándo un sprite ha entrado en colisión con algún carácter del fondo.

Gracias a estos recursos, se pueden escribir programas en BASIC para controlar juegos de movimiento rápido. Lamentablemente, no existen órdenes especiales en BASIC para controlar las configuraciones de los sprites; todo se debe hacer mediante una sucesión de sentencias POKE en la memoria del Commodore 64. Un método alternativo y más sencillo para crear sprites consiste en comprar un cartucho BASIC de Simon.

Basic de Simon
Se trata de un cartucho conectable para ampliar las capacidades de alta resolución y de manipulación de sprites de que dispone el programador en BASIC. El cartucho viene con un voluminoso manual en el que se detallan las 114 órdenes extras. Éstas incluyen órdenes para funcionar en modalidad de alta resolución, para seleccionar los colores del fondo y de primer plano, y para dibujar círculos, elipses, rectángulos y líneas rectas. El manual de instrucciones incluye ayudas para el diseño y la creación de sprites, órdenes para encender y apagar los sprites, y formas de colocarlos en la pantalla



Segundo paso
Estas líneas se pueden agregar al listado del programa de "supermercado" de la p. 359. Esta sección del programa utiliza tres sprites ampliados y multicolores: dos representan la figura humana y otro el carrito de la compra. Los señaladores de datos de los sprites están manipulados para que la forma de la mujer se modifique. Esto da el efecto de una silueta que se desplaza bailando por la pantalla. Para utilizar el programa de "supermercado" como una subrutina de este programa, cambie la línea 3270 para que diga: 3270 RETURN

```
90 REM **SPRITES 64**
100 PRINT "  "
110 V = 53248
120 REM ----LEER DATOS SPRITE----
130 FOR I = 12288TO12350:READA:POKEI,A:NEXT
140 FOR I = 12352TO12414:READA:POKEI,A:NEXT
150 FOR I = 832TO894:READA:POKEI,A:NEXT
160 FOR I = 896TO958:READA:POKEI,A:NEXT
170 FOR I = 12416TO12478:READA:POKEI,A:NEXT
180 REM ----AMPLIAR SPRITES----
190 POKEV+23,7:POKEV+29,7
200 REM ----COLOR SPRITES----
210 POKEV+39,10:POKEV+40,10
220 POKEV+41,1
230 REM ----MULTICOLOR----
240 POKEV+28,3:POKEV+37,7:POKEV+38,9
300 REM ----INDICADORES MEMORIA----
310 POKE2040,192:POKE2041,193:POKE2042,194
320 REM ----ESTABLECER CORDENADAS Y----
330 Y0 = 150:Y1 = Y0+42:Y2 = Y0+34
340 POKEV+1,Y0:POKEV+3,Y1:POKEV+5,Y2
400 REM ----ENCENDER SPRITES----
410 POKEV+21,7
500 GOSUB3000:REM OMITIR A FALTA DE SUBRUTINA
1000 XO = 20
1010 POKE2040,13:POKE2041,14
1020 POKEV,XO:POKEV+2,XO:POKEV+4,XO+48
1030 FORI = 1TO500:NEXT
1040 POKE2040,192:POKE2041,193
1050 XO = XO+5
1060 POKEV,XO:POKEV+2,XO:POKEV+4,XO+48
1070 FORI = 1TO500:NEXT
1080 XO = XO+5
1090 IFXO>200THEN1110
1100 GOTO1010
1110 FORJ = 1TO10
1120 POKE2040,13:POKE2041,14
1130 FORI = 1TO50:NEXT
1140 POKE2040,192:POKE2041,193
1150 FORI = 1TO50:NEXT
1160 NEXT
1170 GOTO1170
9000 REM ----PARTE SUPERIOR MUJER----
9010 DATA0,0,0,21,0,0,21,0,0,22,0,0,86,0
9020 DATA0,86,0,0,86,0,0,40,0,0,252,0
9030 DATA15,255,0,255,255,0,255,255,0
9040 DATA195,255,0,195,255,0,195,243,254
9050 DATA207,243,254
9060 DATA143,240,0,143,252,0,15,252,0
9070 DATA15,252,0,15,252,0
9100 REM ----PARTE INFERIOR MUJER----
9110 DATA15,252,0,15,252,0,15,252,0
9120 DATA15,252,0,5,84,0,5,84,0,5,84,0
9130 DATA5,84,0,10,40,0,234,40,0,234,40,0
9140 DATA234,40,0,192,40,0,192,40,0,0,40,0
9150 DATA0,40,0,0,63,0,0,63,0,0,0,0,0,0
9160 DATA0,0,0
9200 REM ----PARTE SUPERIOR MUJER #2----
9210 DATA0,0,0,0,20,32,32,85,32,32,105,48,48,105,48
9220 DATA48,105,48,48,105,48,48,40,48,48,252,48
9230 DATA63,255,240,63,255,240,63,255,0
9240 DATA3,255,0,3,255,0,3,240,0
9250 DATA15,240,0
9260 DATA15,240,0,15,252,0,15,252,0
9270 DATA15,252,0,15,252,0
9300 REM ----PARTE INFERIOR MUJER #2----
9310 DATA15,252,0,15,252,0,15,252,0
9320 DATA15,252,0,5,84,0,5,84,0,5,84,0
9330 DATA5,84,0,10,40,0,58,168,0,58,168,0
9340 DATA58,0,0,58,0,0,10,0,0,10,0,0
9350 DATA10,0,0,15,192,0,15,192,0,0,0,0,0,0,0
9360 DATA0,0,0
9400 REM ----CARRITO----
9410 DATA192,0,0,224,0,0,118,0
9420 DATA0,55,192,0,32,60,0,53
9430 DATA87,240,32,0,15,53,85,85
9440 DATA32,0,3,53,85,85,0,0,3
9450 DATA21,85,85,31,255,255,24,0
9460 DATA0,12,0,0,12,0,0,31,255
9470 DATA240,31,255,255,1,0,2,7
9480 DATA0,14,7,0,14
```

Osborne-1

Éste es el primer micro portátil que se diseñó y fue también el primero que se vendió con software incluido

Aunque no entra estrictamente en la categoría de ordenador personal, el Osborne-1 es una máquina particularmente interesante porque fue el primer microordenador portátil totalmente autocontenido. Con sus dos unidades de disco incorporadas y su pequeño monitor, el Osborne le ofrece al usuario la posibilidad de trasladar consigo, vaya donde vaya, su propia capacidad para procesamiento de datos. La máquina sólo carece de un paquete de pilas interno, pero el fabricante pensó que ello incrementaría el peso global del ordenador hasta un límite que excedía lo razonable (ya pesa 10,5 kg). Hay, no obstante, un conector de corriente continua en el panel frontal, junto con las otras conexiones para interfaces. La máquina requiere tanto entradas de 12 v como de 5 v: la primera para las unidades de disco y la segunda para la lógica.

El otro factor que incide en contra de su utilización como ordenador personal es su alto precio, aunque en éste se incluyen algunos de los paquetes de software de gestión mejor establecidos que existen, como son: CBASIC Microsoft, una versión compilada del lenguaje BASIC, que posibilita una operación mucho más veloz de los programas; Supercalc, ampliamente reconocido como el mejor de los programas de hoja electrónica de la primera generación; Wordstar y Mailmerge, los paquetes para tratamiento de textos transportables (no limitados a ningún tipo de máquinas) más vendidos; y, quizá lo mejor de todo, el sistema operativo CP/M (Control Program/Monitor), de Digital Research, que permite ejecutar en cualquier máquina que lo utilice una amplia gama de paquetes de software.

El Osborne-1, al igual que el Apple II (véase p. 349), requiere cargar su sistema operativo desde disco. Además de supervisar la operación interna del ordenador, el sistema CP/M es capaz de realizar la mayoría de las labores de conservación: hacer copias de archivos y de discos enteros, iniciar nuevos discos, catalogar el contenido de éstos, etc. Pero el sistema CP/M posee también otros puntos fuertes. En primer lugar, se puede escribir software para el sistema operativo, independientemente de en qué máquina opere. Para la casa fabricante de software esto significa un mercado potencial mucho mayor y por ello se puede invertir muchísimo más dinero en la producción, lo que a su vez asegura un paquete de mayor calidad. En segundo lugar, para un usuario de CP/M experimentado, el tipo de máquina es casi independiente, y esto permite mejorar y perfeccionar el hardware sin la molesta tarea de volver a dar entrada a los archivos de datos y de convertir los programas. Durante un breve período, Osborne incluyó en el precio de compra el dBase II de Ashton-Tate, el más eficaz de todos los programas de administración de bases de datos basados en microordenador.

Unidades de disco de densidad doble
Cada unidad posee una capacidad nominal de 200 Kbytes, que después del formateo se reduce a 184 Kbytes

Microprocesador
El Osborne-1 utiliza el microprocesador Z80A de Zilog, que funciona a 4 MHz

Motorola 6850
Estos chips controlan el funcionamiento de la puerta en serie RS232 estándar

Motorola 6821
Este circuito integrado se emplea para apoyar la puerta de entrada-salida IEEE488 en paralelo

Puerta en serie RS232

Puerta en paralelo IEEE488

Puerta para modem

ROM del sistema

Conector para el teclado



Monitor

ROM para generación de caracteres

Tablero controlador de disco
Conocido técnicamente como "tablero hija" (el tablero de circuito impreso principal se denomina "tablero madre"), desde aquí se controla la operación de la unidad de disco

RAM de 64 Kbytes

Control de brillo del monitor

Control de contraste del monitor

Fuente de alimentación externa

Mando de borrado

Conector para monitor externo

Teclado numérico adicional

Chris Stevens

OSBORNE-1

DIMENSIONES
510 × 325 × 225 mm

PESO
10,5 kg

CPU
Z80A

VELOCIDAD DEL RELOJ
4 MHz

MEMORIA
RAM de 64 Kbytes
ROM de 4 Kbytes

VISUALIZACION EN VIDEO
24 filas de 52 caracteres en una visualización real de 128 × 32

INTERFACES
RS232, IEEE, modem

LENGUAJE SUMINISTRADO
BASIC, Ensamblador Z80

OTROS LENGUAJES DISPONIBLES
Todos aquellos que se ejecuten bajo CP/M.

VIENE CON
CP/M, Wordstar, CBASIC, MBASIC, Mailmerge, Supercalc, Manuales

TECLADO
Estilo máquina de escribir, 69 teclas incluyendo teclado numérico adicional

DOCUMENTACION
Adam Osborne le vendió su empresa editorial a McGraw-Hill con el fin de financiar la fabricación de los ordenadores Osborne, de modo que no es sorprendente que la documentación del manual sea muy buena. El único fallo es la falta de un índice global

Control Program/Monitor

Los ordenadores de unidad principal y los miniordenadores se han beneficiado de los sistemas operativos independientes de la máquina desde que a mediados de los años sesenta se introdujera la segunda generación de ordenadores; no obstante, habrían de transcurrir doce años antes de que los sistemas de control de este tipo llegaran a los microordenadores. El CP/M (Control Program/Monitor), de Digital Research, fue el primero de estos sistemas. Diseñado para los microprocesadores 8080 de Intel y Zilog Z80, posee una gama de programas de uso práctico y domésticos, y define asimismo las formas en que se pueden interrumpir y continuar los programas en ejecución. Otra importante ventaja radica en la definición de las estructuras y los trazados de los archivos, que también manipula el CP/M. Utilizando un programa de intercambio como el BSTAM, que reduce los archivos de cualquier clase a su forma básica, se pueden transferir programas escritos para CP/M entre diversas máquinas, independientemente de su tipo o especificación. Esto significa que el usuario de CP/M dispone de una inmensa cantidad de software

Lamentablemente para Osborne, la mayoría de las empresas comerciales de Estados Unidos concentraron su atención en el ordenador personal de IBM, máquina de 16 bits basada en el microprocesador 8088 de Intel. Concebido como una solución provisional (ostenta un direccionamiento de 16 bits, pero la transferencia de datos es de sólo ocho bits), el 8088 se convirtió en un estándar industrial de facto sencillamente por el hecho de haberlo escogido la IBM para su primera incursión en el mercado de microordenadores.

El ordenador personal IBM utiliza un sistema operativo diseñado especialmente que se denomina PC-DOS. Digital Research lanzó dos nuevas versiones del sistema operativo CP/M: Concurrent CP/M, que permite una verdadera multiprogramación multiusuario, y el CP/M86, diseñado para el chip 8086 de Intel, que incorporaba un direccionamiento de 16 bits y también una transferencia de datos de 16 bits.

Lamentablemente, todos estos desarrollos llegaron demasiado tarde como para impedir que el Osborne-1 se hundiera en los avatares del mercado, y en 1983 la Osborne Computer Corporation (la empresa principal de Estados Unidos) presentó una liquidación voluntaria. Con su memoria de 64 Kbytes (60 Kbytes disponibles para el usuario) y sus unidades de disco gemelas de 183 Kbytes, el Osborne-1 continúa siendo un ordenador razonablemente eficaz. Contribuyen a ello sus puertas RS232 e IEEE incorporadas, la puerta para modem y su capacidad para funcionar con un paquete de pilas, y resulta fácil comprender por qué el ordenador fue un éxito de venta instantáneo y por qué continúa gozando de popularidad entre los usuarios aun después de la desaparición de su primer fabricante.

Una configuración muy interesante del Osborne-1, que hasta cierto punto la comparte con el Epson HX-20 (véase p. 169), es la provisión de una "pantalla virtual" de tamaño más de tres veces mayor que la visualización proporcionada, de 52 columnas por 24 líneas. La utilización de la tecla de control (una exigencia del CP/M estándar) y las teclas del cursor permite que la visualización se mueva a través de la memoria de pantalla real. En gran medida esto elimina la mayoría de los inconvenientes que plantea el pequeño tamaño (8,75×6,6 cm) de la pantalla, aunque quienes no son usuarios de la máquina suelen manifestar su sorpresa ante el hecho de que una visualización cuyos caracteres miden sólo 2 mm de altura pueda ser legible e incluso cómoda de utilizar.

De hecho, son pocos los usuarios que no consiguen adaptarse a esta miniaturización, si bien Osborne proporcionó un conector para monitor externo que reproduce el contenido de la pantalla pequeña en una unidad adicional de mayores dimensiones. En realidad, lejos de considerar que el tamaño de los caracteres era demasiado pequeño, hubo una considerable demanda por parte de los usuarios en el sentido de que toda la pantalla virtual de cuatro Kbytes (128 columnas por 32 filas) se visualizara en todo momento, y Osborne fabricó una modificación para satisfacer esa especificación. Ésta le permite al usuario elegir entre tres anchos diferentes: 52, 96 o 128 caracteres, e incluso en la mayor densidad de caracteres, éstos siguen siendo definidos y legibles.

El teclado del Osborne-1, que se sujeta al panel frontal del ordenador como una "tapa", haciéndolo a prueba de intemperie, es una unidad de 69 teclas. Posee teclas normales estilo máquina de escribir, con la adición de teclas de control CTRL y escape ESC, más un relleno numérico de 12 teclas en el lado derecho que incluye teclas extras de punto y aparte y ENTER. Utilizando un programa CP/M denominado SETUP, las funciones de las teclas numéricas (empleadas conjuntamente con la tecla de control) las puede definir el usuario hasta un máximo de 96 caracteres. Esta configuración es especialmente útil si se ha de usar con frecuencia una palabra, una frase o una orden. Los resultados del programa SETUP se escriben en cada disco, en vez de almacenarse en la memoria, de modo que las funciones se pueden preprogramar por separado para cada paquete diferente de software. El ordenador establece automáticamente sus funciones cada vez que se carga el sistema operativo.

Además de los 96 caracteres estándar de mayúsculas y minúsculas, existen 32 caracteres predefinidos para gráficos, aunque a éstos sólo se puede acceder mediante un programa de aplicaciones.

Debido a que el Osborne-1 utiliza chips de apoyo de la serie 6800 y no sus equivalentes de la familia 8080 (como sería de esperar en una máquina CP/M), el método de sondeo del teclado es ligeramente distinto. Existe una porción de la memoria reservada para interpretar las pulsaciones de las teclas, y la ROM del sistema verifica constantemente si se ha pulsado alguna. En el teclado propiamente dicho no hay ninguna lógica de decodificación. Es esta característica lo que permite una programación sencilla de las teclas de función, y como las funciones se almacenan en la RAM, cabe acceder a ellas y modificarlas desde dentro de un programa.

A pesar de que la Osborne Computer Corporation entró en liquidación voluntaria, la filial británica creó una empresa separada y continuó su actividad comercial. Sea cual fuere el futuro que aguarde a esta máquina, su calidad no admite duda.



Para llevárselo
Además de su excelente relación calidad-rendimiento, el Osborne-1 posee la ventaja adicional de su portabilidad. Con sus 10,5 kg no es un peso ligero, pero es autocontenido y está bien equilibrado, de modo que se transporta sin dificultad.
Una de las condiciones básicas en el momento de precisar sus dimensiones físicas fue su adaptabilidad al limitado espacio disponible debajo del asiento en un avión

Ian McKinnell

Código de clasificación

La clasificación Shell es mucho más eficaz para las matrices largas que la clasificación burbuja o por inserción. Funciona dividiendo los datos en una serie de "cadenas"

En la p. 286 analizamos dos métodos para clasificar por orden una matriz: las clasificaciones burbuja y por inserción. Por lo general, la clasificación burbuja es más fácil de realizar, pero la clasificación por inserción es más rápida. La experiencia con estos dos procedimientos demuestra que lo que más tiempo ocupa es cambiar los naipes de un lugar a otro a través de distancias cortas: suele ser preferible cambiar un naipe de lugar una sola vez a través de una distancia larga que cambiarlo varias veces en desplazamientos cortos.

```
7999 REM * * * * * * * * * *
8000 REM *      SHELL       *
8001 REM * * * * * * * * * *
8025 PRINT "CLASIFICACION SHELL - ADELANTE!!!!!"
8050 LET LK = LT
8100 FOR Z = O TO I STEP O
8150 LET LK = INT(LK/II)
8200 FOR LB = I TO LK
8250 LET LL = LB+LK
8300 FOR P = LL TO LT STEP LK
8350 LET D = R(P)
8400 FOR Q = P TO LL STEP-LK
8450 LET R(Q) = R(Q-LK)
8500 IF D< = R(Q) THEN LET R(Q) = D: LET Q = LL
8550 NEXT Q
8600 IF D>R(LB) THEN LET R(LB) = D
8650 NEXT P
8700 NEXT LB
8750 IF LK = I THEN LET Z = I
8800 NEXT Z
8850 PRINT "CLASIFICACION SHELL - STOP!!!!!"
8900 RETURN
```

Para agregar esta rutina al programa de demostración de clasificación de la p. 287, cambie la línea 350 por:

```
350 LET I = 1: LET O = 0: LET II = +: LET TH = 3
```

y cambie la línea 900 por:

```
900 ON SR GOSUB 6000,7000,8000
```

Un procedimiento mejor que estos dos es el denominado *clasificación Shell* (llamado así en honor de su creador, D. Shell). Este método disminuye el desorden de la matriz al comenzar la clasificación (de modo que los datos no están muy lejos de sus posiciones verdaderas) y permite que las permutaciones operen a través de distancias relativamente grandes. He aquí un método para esta clasificación:

1) Disponga los naipes de un mismo palo en cualquier orden. Éstos habrán de clasificarse por orden descendente, de manera que el Rey será el naipe situado más hacia la izquierda y el As el situado más a la derecha. Cuente los naipes, divida el número (en este caso, 13) por dos, ignorando la cantidad que sobre, y escriba el resultado (es decir, 6) en un trozo de papel bajo el título "el eslabón".

2) Coloque una moneda de una peseta debajo del naipe situado más a la izquierda (a ésta la llamará posición 1) y una moneda de cinco pesetas en la posición eslabón (es decir, en posición 6 en el primer caso). Todos los naipes desde la posición 1 hasta la posición eslabón serán los naipes situados al extremo izquierdo de una serie de "cadenas" de cartas. El número de cadenas será igual al valor en curso del eslabón. Cada cadena se forma empezando por el naipe situado más a su extremo, sumando el eslabón al número de posición del naipe del extremo para obtener la posición del naipe siguiente, y así sucesivamente hasta que se alcanza o se supera el extremo de la matriz. La primera cadena comprende los naipes de las posiciones 1, 7 y 13; la segunda cadena consta de los naipes de las posiciones 2 y 8; la tercera, de los de las posiciones 3 y 9. La última es la de los naipes de las posiciones 6 (el valor actual del eslabón) y 12.

3) Ahora, después de haber marcado los límites con las monedas de una y de cinco pesetas, desplace los naipes que constituyen la primera cadena fuera de la matriz de modo que usted los pueda ver en solitario, y póngalos en orden utilizando ya sea la clasificación burbuja o la clasificación por inserción tal como las describimos en la p. 286.

4) Vuelva a desplazar la cadena ordenada de nuevo en los espacios de la matriz y repita el punto anterior con la siguiente cadena, y después con la siguiente, hasta que se hayan clasificado todas las cadenas cuyos naipes más sobre la izquierda estén dentro de las monedas de una y cinco pesetas.

5) Cuando haya clasificado todas las cadenas, divida el eslabón por dos. Si éste es menor que uno, la matriz está ordenada. Si no, repita el proceso desde el paso 2 con el nuevo valor del eslabón.

Panel de clasificación Shell

Posición n.º									Valor eslabón	Comentarios
1	2	3	4	5	6	7	8	9		
2	8	9	3	D	5	K	6	7	(9/2)=>4	Empezar pasada
*	+	@	$	*	+	@	$	*		Formar cadenas
D				7				2		Clasif. cadena 1
	8				5					Clasif. cadena 2
		K				9				Clasif. cadena 3
			6				3			Clasif. cadena 4
D	8	K	6	7	5	9	3	2		Empezar pasada
D	8	K	6	7	5	9	3	2	(4/2)=>2	Fin pasada
*	+	*	+	*	+	*	+	*		Formar cadenas
K		D		9		7		2		Clasif. cadena 1
	8		6		5		3			Clasif. cadena 2
K	8	D	6	9	5	7	3	2		Fin pasada
K	8	D	6	9	5	7	3	2	(2/2)=>1	Empezar pasada
*	*	*	*	*	*	*	*	*		Formar cadena 1
K	D	9	8	7	6	5	3	2		Fin pasada

CLAVE	
*	Miembro cadena 1
+	Miembro cadena 2
@	Miembro cadena 3
$	Miembro cadena 4

Clasificación Shell
El ejemplo de la clasificación Shell que vemos en el papel, para una mano reducida, demuestra su método exclusivo de dividir la matriz en una serie de cadenas (con espaciaciones basadas en el número de eslabón corriente). Estas cadenas se clasifican por separado, en este caso utilizando el método por inserción, antes de que se complete una pasada.
El listado del programa que aquí damos para una clasificación Shell se debe emplear conjuntamente con el programa *testbed* de la p. 287. Cuando lo probamos, se observó una significativa mejora respecto a los otros métodos de clasificación cuando el número de ítems a clasificar superaba los 40

Con una sola mano

El Microwriter es un procesador de textos portátil que se puede manejar con una sola mano. El teclado de seis botones quizá se utilice pronto en los ordenadores

Tener un procesador de textos en la oficina, o un ordenador personal con un programa para tratamiento de textos, puede ser una buena idea. Aparte de eliminar el fatigoso trabajo de escribir cartas o documentos rutinarios, puede ayudar con la documentación de programas, elaborar con toda rapidez copias de avisos o cuidar de una agenda de direcciones. Sin embargo, el problema surge cuando usted desea llevarse notas de su casa o de la oficina en una forma que el ordenador pueda comprender.

Existe un mercado en crecimiento para los sistemas informáticos portátiles como el Tandy 100 y el Epson HX-20. Si bien éstos poseen la ventaja de que pueden actuar como procesadores de textos portátiles, o como terminales remotos de sistemas mayores, no son tan manuales como un bloc de notas o un dictáfono. ¿Qué diría usted de un sistema de tratamiento de textos que fuera lo suficientemente pequeño como para llevarlo en el bolsillo? Un sistema que fuera tan compacto como para funcionar a pilas y que se pudiera manejar con una sola mano, pudiéndose conectar con una impresora o incluso con otro ordenador.

Hace aproximadamente cuatro años que existe este tipo de dispositivo y se denomina Microwriter. Concebido originalmente por el norteamericano Cy Enfield, cambia el teclado QWERTY en favor de un sistema exclusivo de teclas múltiples con sólo seis teclas de botón. El concepto surgió originalmente del deseo de crear un juego manual basado en palabras, para el cual hasta un teclado en miniatura hubiera resultado demasiado grande y demasiado caro. La respuesta evidente era la creación de un tipo especial de teclado que utilizaba sólo unas pocas teclas con suficientes combinaciones para especificar todos los símbolos alfanuméricos. El sensacional avance se produjo con la invención de un sistema de código simbólico que es exclusivo del Microwriter.

A primera vista parece imposible que las letras del alfabeto, por no hablar de los símbolos numéricos y los signos de puntuación, se puedan crear a partir de combinaciones entre apenas seis teclas, pero éstas son más que suficientes. Y aprenderse todas las combinaciones comunes lleva apenas unas pocas horas. En realidad, como con cierta justificación afirman los fabricantes, es mucho más fácil de aprender que un teclado QWERTY. La combinación de teclas necesaria para cada una de las letras se basa en la forma física de la letra, un código que suele resultar más fácil de aprender a quienes no saben mecanografía. Debido a que sólo se necesita una mano, el Microwriter también les abre las puertas del tratamiento de textos a los minusválidos, que no pueden manipular las múltiples pulsaciones de teclas que a menudo se requieren para generar órdenes en un teclado convencional.

Interface para cassette
Funciona con una grabadora doméstica

Puerta para salida
Esta puerta proporciona una interface RS232 para una impresora, un ordenador o un acoplador acústico. Con un adaptador externo también puede visualizar en un monitor o un televisor

Visualización en cristal líquido
A pesar de configurar sólo 16 posiciones de caracteres, por razones de legibilidad los caracteres se forman en una gran matriz

Microinterruptores
Estos dispositivos se utilizan para atemperar la presión necesaria para activar los botones

RAM
La máquina viene con 8 K estándar, pero en los mismos conectores se pueden incorporar chips más grandes para aumentar esta capacidad



Internamente, el Microwriter está diseñado para ser lo más portátil posible. Tanto el microprocesador interno como su memoria son dispositivos CMOS (Complementary Metal Oxide Silicon) que ayudan a reducir el consumo energético. Un paquete de pilas níquel-cadmio recargable proporciona la energía suficiente para 30 horas de uso. Para visualizar los caracteres, hay una LCD de 14 caracteres (que gira horizontalmente a medida que se va dando entrada al texto), incorporada en la unidad, aunque también podemos conectar un monitor a través de una interface opcional. Esto permite obtener la edición en pantalla del texto almacenado una vez que el usuario ha regresado al hogar o a la oficina.

Además de contar con una interface en serie RS232 para conectar a una impresora, el Microwriter está equipado con una interface para cassette, que permite guardar permanentemente o volver a cargar el texto almacenado en su memoria. Asimismo, la interface en serie permite que el Microwriter se pueda utilizar como una terminal manejable con una sola mano de un ordenador o un procesador de textos normales. Los documentos digitados fuera del hogar o la oficina se pueden cargar en un sistema de tamaño natural para una edición o una manipulación más completas.

El texto dentro del Microwriter se puede dividir en documentos separados, y ello permite dar entrada o manipular por separado diversos bloques de texto sin relación entre sí. Existen sencillos resortes para agregar o eliminar texto y hasta desplazar grandes bloques utilizando una cassette como memoria intermedia.

Fue intención del diseñador que el teclado de seis teclas del Microwriter pudiera incorporarse a otros dispositivos electrónicos. No obstante y a pesar de sus muchas cualidades, el Microwriter sólo ha tenido una aceptación limitada y aún está por ver si los fabricantes de ordenadores personales se hacen eco de la idea.

Interruptor
Al encenderlo o apagarlo no se pierden datos y se puede continuar con la escritura del mismo documento

Cristal del reloj

Enchufe red
Para recargar o para funcionar en conexión a la red eléctrica con un transformador externo

CPU
Tanto la CPU como la RAM son dispositivos CMOS (Complementary Metal Oxide Silicon) para reducir el consumo energético

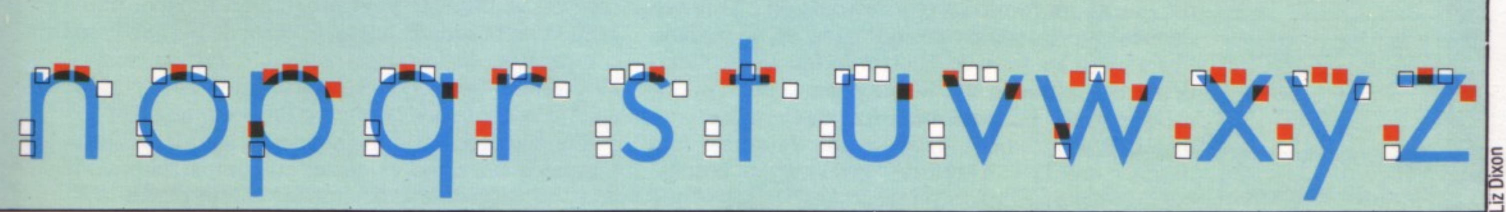
Pilas de Ni-Cad
Estas pilas de níquel-cadmio se recargan mediante un transformador externo

Interface para ampliación
En vistas a una futura ampliación, esta puerta incluye las líneas de dirección y de datos del microprocesador

EPROM
El programa de tratamiento de textos y el sofisticado software de comunicaciones están incorporados en una única EPROM, cuya fabricación en pequeñas cantidades resulta más económica que una ROM

Steve Cross

Grupos de cinco
La documentación del Microwriter incluye notas mnemotécnicas e ilustraciones para ayudar al usuario a aprender las distintas combinaciones de teclas necesarias para crear el alfabeto. La sexta tecla se utiliza en combinación con las otras para proporcionar las órdenes de puntuación y edición



Búsqueda binaria

El tiempo necesario para localizar un registro se reduce mucho merced a la "búsqueda binaria", si el archivo ya se ha clasificado en el orden adecuado

Las tres actividades más importantes del programa de la agenda de direcciones (agregar registros nuevos, guardar el archivo en cinta o disco y leer el archivo de datos cuando se ejecuta por primera vez el programa) ya las hemos desarrollado. Pero una agenda de direcciones no sirve de nada si usted sólo puede agregar información irrecuperable. Se necesita, pues, una rutina para hallar un registro.

Probablemente la actividad más frecuente será hallar el registro completo de un nombre, y ése es el motivo por el cual la primera opción del menú de opciones (*ELECCN*) es HALLAR REGISTRO (DE UN NOMBRE). Buscar es una actividad muy importante en la mayoría de los programas para ordenadores, en especial en los programas para bases de datos donde a menudo se necesita recuperar datos de un archivo. En términos generales, existen dos métodos de búsqueda: el lineal y el binario. El lineal observa cada uno de los elementos de una matriz, empezando desde el principio, y continúa hasta encontrar el dato deseado. Si los datos de la matriz están sin clasificar, la única búsqueda que ofrece garantías de funcionamiento es la lineal. El tiempo necesario para localizar el dato utilizando una búsqueda lineal en una matriz de N datos posee un valor promedio proporcional a N/2. Si son pocos los datos a través de los cuales se debe buscar, N/2 puede ser perfectamente aceptable; pero a medida que aumenta el número de datos, el tiempo que ocupa realizar la búsqueda también aumenta, hasta resultar excesivo.

No obstante, si se sabe que los datos del archivo ya están clasificados, existe un método de búsqueda mucho más eficaz que se denomina *búsqueda binaria*, y funciona de la siguiente manera. Supongamos que se desea hallar la definición de la palabra "lepidóptero" en el diccionario. Desde luego no comenzará por la primera página a ver si está allí y, de no hallarla, pasará a la segunda página y así sucesivamente a través de todo el diccionario hasta que encuentre la palabra deseada. Lo que hará es abrir el libro aproximadamente por la mitad, elegir una página y mirar lo que hay en ella. Si resulta que comienza por la palabra "metatarso", usted sabe que se ha pasado, de manera que la segunda mitad del libro es descartada, pues la palabra que busca está en algún lugar de la primera mitad. Volverá entonces a repetir el proceso, considerando ahora la página por donde abrió el diccionario la primera vez como si fuera la última página. Vuelve a dividir en dos esta parte del diccionario y a escoger una página, que ahora comienza con "involución". Esta vez deduce que la página seleccionada es demasiado "baja" y (a los fines de nuestra búsqueda de "lepidóptero") se puede considerar como si fuera la primera página: las anteriores son descartadas y se dice que son "bajas" en el sentido de que la *l* va después que la *i*. Ahora de "primera" y "última" páginas del diccionario pueden hacer aquellas que empiezan con "involución" y "metatarso" respectivamente. De nuevo abre por la mitad de la sección restante y encuentra "juncal" iniciando página. Una vez más es demasiado "baja", de modo que la palabra que estamos buscando debe estar entre ésta y la página de "metatarso". La suficiente repetición de este proceso es una garantía para localizar la palabra que estamos buscando (¡siempre que esté en el diccionario!).

En el ejemplo que acabamos de considerar, "lepidóptero" era la *clave de búsqueda*. La clave de búsqueda es la entrada que estamos intentando hallar. Cada vez que examinamos un registro, comparamos la clave de búsqueda con la *clave de registro* para localizar el "objetivo" o "víctima". Junto con la clave de registro esperamos hallar lo que se denomina *información adicional*, con la suficiente lógica. La información adicional para la clave de registro "lepidóptero" sería la definición que da el diccionario para la palabra, en este caso, insecto llamado corrientemente "mariposa".

La analogía es perfecta con la búsqueda de un registro-objetivo a través de un archivo en una base de datos, siempre y cuando los registros se hayan clasificado previamente como se han clasificado las entradas de un diccionario. ¡Piense lo difícil que sería utilizar un diccionario si las entradas estuvieran consignadas por el orden en que se le hubieran ocurrido al lexicógrafo!

La rutina de búsqueda requerida para nuestra agenda de direcciones resultará algo más complicada de lo que podríamos imaginar en un primer momento, por motivos que más adelante se harán evidentes. Lo primero que hará la rutina de búsqueda (de momento la llamaremos *BUSREG) será solicitar el nombre que ha de hallar. Es la clave de búsqueda. Supongamos que en algún lugar del archivo hay un registro para una persona llamada María Gómez. El registro para esta persona tendrá un campo (con el nombre en forma estandarizada), que contenga GOMEZ MARIA. La rutina de búsqueda podría presentarse con un mensaje como ¿A QUIEN ESTA BUSCANDO? y nosotros le responderíamos MARIA GOMEZ, o quizá M. GOMEZ o Mari Gómez. Antes de que todo esto se complique demasiado, vamos a suponer que nosotros le respondemos con el nombre completo, María Gómez. Lo primero que hará la rutina de búsqueda será adaptar esta respuesta a la forma estandarizada, GOMEZ MARIA. A continuación, comparará nuestra entrada, la clave de búsqueda, con los diversos contenidos de los campos MODNOM$. Si el programa estuviera utilizando una búsqueda lineal, la clave de búsqueda se compararía, de forma secuencial, con cada campo MODNOM$, hasta que se encontrara un nombre igual o se descubriera que no hay otro igual.

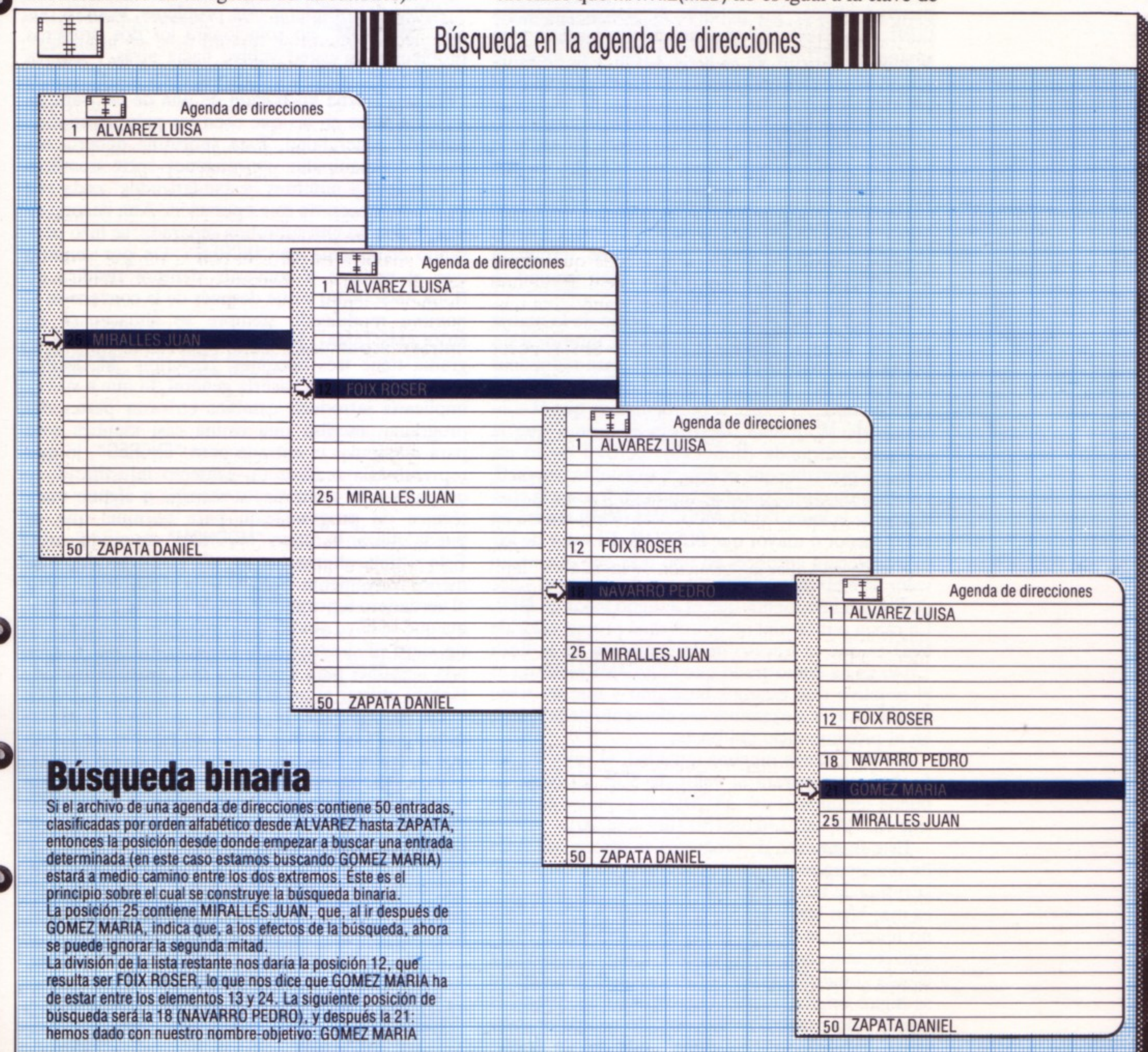
Sin embargo, como ya hemos apuntado, una búsqueda lineal no es eficaz en comparación con una búsqueda binaria cuando los datos ya están cla-

sificados. La rutina de búsqueda puede asegurar que los registros estén clasificados empezando con una IF RMOD = 1 THEN GOSUB *BUSREG*. El programa sabe que el elemento más bajo de la matriz a buscar será MODCAM$(1) y que el más alto será MODCAM$(TAMA-1). Para conducir la búsqueda necesitaremos tres variables: INF, para la parte inferior de la matriz (MODCAM$(1) al principio); SUP, para la parte superior de la matriz (MODCAM$(TAMA-1) al principio); y MED, para el valor correspondiente al elemento medio.

Ahora, utilizando de nuevo la analogía del diccionario, podemos suponer que INF = MATRIZ(1) y SUP = MATRIZ(TAMA-1). En otras palabras, la matriz que hemos considerado para la búsqueda empieza por el elemento "más grande". Por consiguiente, podemos dejar que LET INF = 1 y LET SUP = TAMA-1 (recuerde que TAMA siempre es mayor en uno que el número de registros que existe habitualmente en la agenda de direcciones).

Supongamos que en la agenda de direcciones hay 21 entradas válidas. TAMA tendrá un valor de 22. INF tendrá un valor de 1. SUP tendrá un valor de 21. El valor de MED, es decir, la posición del elemento medio, se puede obtener en BASIC mediante INT((INF + SUP)/2); si el valor de INF es 1, y el valor de SUP es 21, el valor de MED será, entonces, 11.

Para realizar una búsqueda binaria, primero suponemos que todo el archivo es válido y hallamos el punto medio INT(INF + SUP)/2) dentro de un bucle que se termina si se ha hallado el objetivo o bien si no existe igualdad. Luego verificamos para ver si la clave de búsqueda (BUSCLV$) resulta ser igual al valor MED de la matriz. Si el valor MED de la matriz es demasiado pequeño, sabemos que MATRIZ(MED) es la parte más baja de la matriz que necesitamos considerar, de modo que INF se podría establecer en MED. Sin embargo, es ligeramente más eficaz establecer INF en MED + 1, dado que ya sabemos que MATRIZ(MED) no es igual a la clave de



Búsqueda binaria
Si el archivo de una agenda de direcciones contiene 50 entradas, clasificadas por orden alfabético desde ALVAREZ hasta ZAPATA, entonces la posición desde donde empezar a buscar una entrada determinada (en este caso estamos buscando GOMEZ MARIA) estará a medio camino entre los dos extremos. Este es el principio sobre el cual se construye la búsqueda binaria.
La posición 25 contiene MIRALLES JUAN, que, al ir después de GOMEZ MARIA, indica que, a los efectos de la búsqueda, ahora se puede ignorar la segunda mitad.
La división de la lista restante nos daría la posición 12, que resulta ser FOIX ROSER, lo que nos dice que GOMEZ MARIA ha de estar entre los elementos 13 y 24. La siguiente posición de búsqueda será la 18 (NAVARRO PEDRO), y después la 21: hemos dado con nuestro nombre-objetivo: GOMEZ MARIA

búsqueda. Del mismo modo, por otra parte, SI MATRIZ(MED) > BUSCLV$, SUP se establecería en MED − 1.

Como paso provisional hacia el desarrollo de una rutina que funcione por completo, el programa mostrado puede tomar una entrada ficticia (que necesita estar exactamente en el mismo formato que los campos de MODCAM$), o bien imprimir NO HALLADO REGISTRO si no hubiera encontrado igualdad o EL NUMERO DE REGISTRO ES (MED) si la hubiera hallado. Como la rutina empieza con el número de línea 13000, se puede agregar al final del programa tal como lo presentamos en la p. 399, y funcionará siempre que la línea 4040 se cambie por IF OPCN = 1 THEN GOSUB 13000.

La línea 13240 contiene la sentencia STOP. Ésta interrumpirá temporalmente el programa tan pronto como se visualicen los mensajes NO HALLADO REGISTRO o EL NUMERO DE REGISTRO ES (MED). El programa se puede reemprender por el mismo número de línea, sin pérdida de datos, digitando CONT. Sin STOP, el programa se apresuraría a la sentencia RETURN de la línea 13250 y el mensaje aparecería durante un lapso demasiado breve como para resultar legible.

Consideremos con mayor detalle este fragmento de programa. La línea 13100 establece INF en 1, la posición del menor elemento de la matriz MODCAM$. SUP se establece en TAMA-1 en la línea 13110. Ésta es la posición de las matrices MODCAM$ donde está situado el elemento mayor. La línea 13120 inicializa un bucle que sólo se terminará cuando se encuentre una igual o bien cuando se sepa que no existe igualdad alguna.

La línea 13130 halla el punto medio de la matriz dividiendo por dos la suma del índice inferior y superior de la matriz (se emplea INT para redondear la división, de modo que MED no pueda asumir un valor como 1,5). Existe la posibilidad de que el contenido de MODCAM$ (MED) sea el mismo que la clave de búsqueda (BUSCLV$); pero si no lo es, como es probable que ocurra, L se establecerá en 0, asegurando que el bucle se repita. Si fracasa la condición de la línea 13140, MODCAM$ (MED) tendrá un valor menor o mayor que BUSCLV$. El valor de INF se establecerá entonces en uno más que el antiguo valor de MED (línea 13150), o se establecerá el valor de SUP en uno menos que el antiguo valor de MED. La razón por la cual no se utiliza el propio valor de MED es que el fracaso de la condición de la línea 13140 ya ha demostrado que MODCAM$(MED) no es el objetivo que estamos buscando y que no tiene sentido volver a examinar ese elemento de la matriz en el próximo ciclo del bucle.

De no hallarse ninguna igualdad, el valor de INF finalmente superará al valor de SUP. El bucle se puede terminar (línea 13170) e imprimirse el mensaje NO HALLADO REGISTRO (línea 13200).

Este fragmento de programa se ofrece con fines de demostración y para permitir la verificación de la rutina de búsqueda. Tal como está, su utilización es más bien limitada. Sin el STOP de la línea 13240 no tendríamos tiempo ni siquiera para ver el mensaje centelleando en la pantalla. Lo que se necesita es una visualización del registro completo, tal como se digitó originalmente. Una vez que se conoce el número del registro resulta sencillo recuperar cualquier información adicional que se necesite: NOMCAM$, CLLCAM$, etc. Debajo de la visualización del registro probablemente desearíamos un mensaje como PULSE BARRA ESPACIADORA PARA CONTINUAR (regreso al menú principal) y quizá otras opciones como PULSE "I" PARA IMPRIMIR

Lamentablemente, ya no es tan fácil decidir cómo manipular la entrada de *ENCREG*. En el fragmento de programa, la entrada esperada (en la línea 13020) ha de estar en la forma estandarizada: GOMEZ MARIA, por ejemplo. Esto, sin duda, no es todo lo conveniente que sería de desear. La gente no piensa los nombres en orden inverso, y tener que darles entrada en letras mayúsculas representa una incomodidad para el usuario. Además, a la más mínima desviación respecto al nombre al que originalmente se dio entrada nos encontraríamos con NO HALLADO REGISTRO. Cabría esperar que los dos primeros problemas los manipulara *MODNOM*. El tercer problema, el de cómo hacer frente a una pareja aproximada, es muchísimo más interesante pero, al mismo tiempo, mucho más difícil de resolver.

Antes de considerar este problema, veamos por qué *MODNOM* no solucionará los dos primeros problemas. Si usted vuelve hacia atrás y analiza *MODNOM*, que empieza en la línea 10200, descubrirá una buena ilustración de una de las trampas más comunes en las que caen los programadores: falta de generalidad. Esta subrutina debería ser capaz de manipular conversiones entre nombres "normales" y nombres "estandarizados" cada vez que fuera necesaria esta operación. Aun cuando se escribió como una subrutina separada, se hizo pensando claramente en *INCREG*. Da por sentado que el nombre a convertir siempre residirá en NOMCAM$(TAMA) y que después de la conversión el nombre modificado siempre se almacenará en MODCAM$(TAMA). Frente a esta situación, el programa tiene tres opciones: reescribir *MODNOM* por completo para hacerla general, lo que a su vez implicaría introducir cambios en otras partes del programa; escribir una rutina casi idéntica sólo para manipular la entrada para *ENCREG*, lo que representaría realizar un esfuerzo injustificado y ocuparía más memoria; o recurrir a alguna mala técnica de programación para permitir que se pueda utilizar la rutina *MODNOM* sin modificarla. Esta última alternativa es, en varios sentidos, la menos conveniente. Solucionaría el problema, pero el verdadero funcionamiento de la parte del programa que se haya modificado probablemente no estaría claro ni siquiera para el programador, y sería una pesadilla para cualquier otra persona que intentara utilizar el programa.

La moraleja de la historia es: hacer que las subrutinas sean lo más generales posible, de modo que cualquier parte del programa las pueda llamar.

Para ilustrar una técnica de programación mala o, como a menudo se la suele llamar, programación "chapuza", y para mostrar lo poco claro que puede hacer que resulte el programa, consideremos la línea 13020 del fragmento de programa INPUT "DE ENTRADA A LA CLAVE"; BUSCLV$ y analicemos después la modificación o el "arreglo" que posibilitaría la utilización de *MODNOM*:

```
13020 INPUT "DE ENTRADA A LA
      CLAVE";NOMCAM$(TAMA)
13030 GOSUB 10200: REM SUBRUTINA
      *MODNOM*
13040 LET BUSCLV$ = MODCAM$(TAMA)
13050 ...
```

Afortunadamente, el valor de TAMA siempre es mayor en uno que el más alto de los registros válidos. En otras palabras, en las matrices no hay ningún registro en la posición TAMA, de manera que este arreglo no modificará ninguno de los registros existentes. Pero sin algunas otras observaciones REM que expliquen lo que está sucediendo, ¡imagínese lo confusas que le resultarían estas tres líneas a alguien que no hubiera estado implicado en el desarrollo del programa!

Volvamos a un problema más interesante, el de tratar con los "cuasi yerros". Supongamos que hemos dado entrada al nombre de alguien como Mari Gómez durante una operación de *INCREG*, pero como María Gómez durante *ENCREG*. Éstos se convertirían a las formas estandarizadas GOMEZ MARI y GOMEZ MARIA, respectivamente, y durante la búsqueda no se encontraría ningún emparejamiento, incluso aunque el registro que deseáramos estuviera allí. Nosotros no intentaremos resolver este problema, porque una solución satisfactoria representaría un gran trabajo de programación. No obstante, he aquí algunas indicaciones para aquellos lectores interesados en experimentar:

```
EMPEZAR {buscar matriz para pareja exacta}
  IF se halla igualdad
    THEN PRINT registro completo
    ELSE buscar matriz para pareja aproximada
      IF se halla pareja aproximada
        THEN PRINT registro de pareja aproximada
        ELSE PRINT "NO HALLADO REGISTRO"
      ENDIF
  ENDIF
END
```

El procedimiento para pareja aproximada se podría parecer a las siguientes líneas:

```
EMPEZAR pareja aproximada
  Buscar matriz para igualdad de apellido
  IF igualdad de apellido
    THEN buscar nombres de pila aproximados
    PRINT registro aproximado
    ELSE buscar apellidos aproximados
      IF se halla apellido aproximado
        THEN PRINT registro aproximado
      ENDIF
  ENDIF
END
```

El procedimiento para "aproximados" se podría definir en líneas generales como hallar la variable objetivo que posea el máximo número de caracteres en común con aquellos de la variable clave. Podría aceptar una situación en la que la variable clave estuviera totalmente contenida en la variable objetivo, o viceversa. No existen soluciones sencillas, pero el panorama es muy amplio para una programación emprendedora.

En el fragmento de programas presentado existe un "efecto colateral". Supongamos que se produjera la siguiente secuencia de acontecimientos. Usted ejecuta el programa y después utiliza *INCREG* para agregar un registro nuevo, seguido de *ENCREG* para localizar un registro. Cuando por último se ejecute *SAPROG*, para guardar el archivo y dar por terminado el programa, no se guardará el registro que usted agregó (aunque sí se guardarán todos los otros registros). Esto es consecuencia directa de algo que sucedió durante la ejecución de *ENCREG*. ¿Se da cuenta por qué no se guardará el registro agregado?

En el próximo capítulo de nuestro curso explicaremos cómo evitar esta pérdida de datos; mostraremos para qué se utiliza la variable CURS y describiremos cómo borrar o modificar un registro. Otras opciones del menú principal (*ENCCIU*, etc.) son bastante similares a las rutinas que ya hemos elaborado. Se deja a los propios lectores su creación, en el caso de que sean necesarias.

Por último, veamos qué sucedería si hubiera 50 registros en el archivo de datos y se usara la rutina *ENCREG* modificada (que llama a *MODNOM*). (Una pista: TAMA tendrá el valor 51.)

Complementos al BASIC

SPECTRUM

Modificaciones para el Spectrum:

```
13000 REM VERSION PARA PROBAR
      *ENCREG*
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT "DE ENTRADA A LA
      CLAVE";B$
13100 LET INF = 1
13110 LET SUP = TAMA - 1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF + SUP)/2)
13140 IF M$(MED) <> B$ THEN LET L = 0
13150 IF M$(MED) < B$ THEN LET
      INF = MED + 1
13160 IF M$(MED) > B$ THEN LET
      SUP = MED - 1
13170 IF INF > SUP THEN LET L = 1
13180 NEXT L
.
.
.
13200 IF INF > SUP THEN PRINT "NO
      HALLADO REGISTRO"
13210 IF INF < = SUP THEN PRINT "EL
      NUMERO DE REGISTRO ES ";MED
.
.
.
13240 STOP
13250 RETURN
```

De nuevo el problema de las variables alfanuméricas de una letra en el Spectrum: aquí B$ se ha sustituido por BUSCLV$.

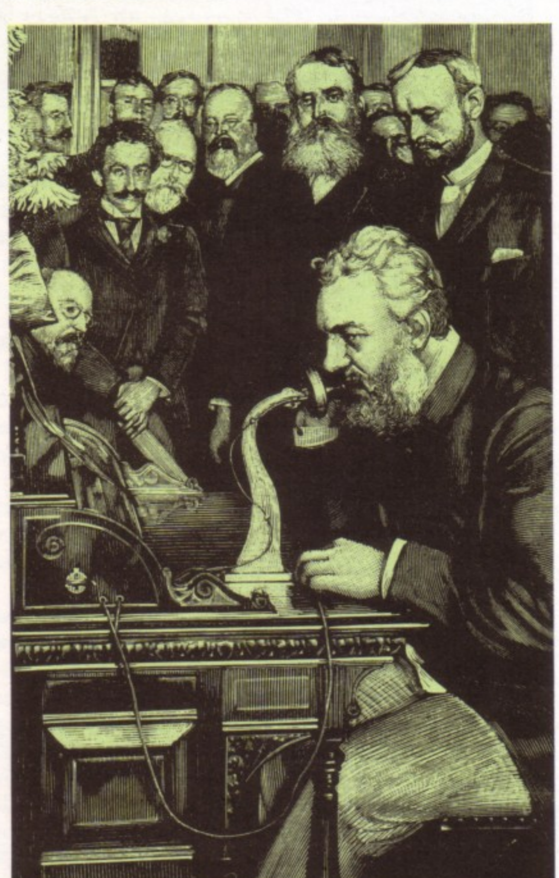
```
13000 REM VERSION DE *ENCREG* PARA PROBAR
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT "DE ENTRADA A LA CLAVE";BUSCLV$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET INF = 1
13110 LET SUP = TAMA-1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF+SUP)/2)
13140 IF MODCAM$(MED)<>BUSCLV$ THEN L = 0
13150 IF MODCAM$(MED)<BUSCLV$ THEN INF = MED+1
13160 IF MODCAM$(MED)>BUSCLV$ THEN SUP = MED-1
13170 IF INF>SUP THEN L = 1
13180 NEXT L
13190 REM
13200 IF INF>SUP THEN PRINT "NO HALLADO REGISTRO"
13210 IF INF<=SUP THEN PRINT "EL NUMERO DE REGISTRO ES ";MED
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

Los Laboratorios Bell

En la historia del ordenador, este centro de investigación ha realizado numerosas aportaciones, tanto en el campo del hardware como en el del software

Hace cien años, la reina Victoria quedó encantada con un nuevo invento que le permitía hablar desde la isla de Wight con sus ministros, en Londres. El teléfono ha sido objeto de notables mejoras desde aquellos tiempos del aparato a manivela a través de la investigación y el desarrollo, y una de las consecuencias de este trabajo ha sido el ordenador. En los primeros años de la historia del teléfono, la American Telephone and Telegraph Company decidió crear una organización que investigara la forma de perfeccionar el sistema telefónico. Así nacieron en 1925 los Laboratorios Bell (apodados *Ma Bell*) en Murray Hill, Nueva Jersey (Estados Unidos).

Laboratorios Bell es una institución insólita porque se dedica exclusivamente a la investigación y, sin embargo, es propiedad de una compañía con fines lucrativos. A los científicos se les mantiene deliberadamente alejados de los problemas de ingeniería que plantea día a día el funcionamiento de una empresa de estas características, porque la Bell considera que la investigación supone una inversión rentable a largo plazo. Resumiremos aquí algunos aspectos de su labor, los más decisivos para el desarrollo del ordenador.



"¡Venga aquí, señor Watson: lo necesito!"
Los Laboratorios Bell han tomado su nombre de Alexander Graham Bell (1847-1922), a quien se atribuye la invención del teléfono, en el año 1876. Se dice que las primeras palabras que se transmitieron por medios eléctricos a través de cables fueron las que dirigió Bell a su ayudante, que estaba en el cuarto contiguo; dichas palabras fueron: *"Come here, Mr. Watson, I want you!"* ("¡Venga aquí, señor Watson: lo necesito!")

A mediados de la década de los treinta, los sistemas telefónicos se iban automatizando y haciéndose cada vez más complejos. Los mensajes se enviaban en modo analógico a través de los cables telefónicos y las llamadas se conectaban utilizando la información contenida en un código de discado digital. El número discado se convertía primero, en la central telefónica, de una señal analógica a una secuencia de impulsos digitales. Ésta se almacenaba temporalmente en una memoria compuesta por interruptores de relé hasta completar la conexión mediante un banco de interruptores en cruz. Éstos contaban los impulsos del código de discado y los convertían en coordenadas sobre un cuadro conmutador electromecánico. Todos los ingredientes de un ordenador ya estaban ahí: sólo esperaban a que la persona adecuada los recogiera.

George Stibitz, matemático que trabajaba en la Bell, observó la similitud entre los impulsos "contadores" y la suma de todos ellos juntos. Trabajando en su casa, sobre la mesa de la cocina, con algunos antiguos interruptores en cruz y relés electromecánicos, construyó los primeros circuitos de ordenador de relé.

Stibitz comenzó entonces a trabajar con un experimentado ingeniero de interruptores, Samuel B. Williams, que llevaba 25 años construyendo circuitos interruptores, y los dos hombres crearon una calculadora de números complejos (los números complejos comprenden los llamados números imaginarios, raíces cuadradas de números negativos, y son necesarios para obtener soluciones completas de una ecuación polinómica). Comenzaron la tarea en 1937, y el dispositivo consumió 450 relés y 10 interruptores en cruz. Operaba en notación binaria y podía dividir dos números de ocho dígitos en 30 segundos. La calculadora de números complejos entró en funcionamiento el 8 de enero de 1940, y en septiembre del mismo año se exhibió ante la American Mathematical Society, en el Dartmouth College (donde posteriormente se idearía el BASIC). La calculadora tenía capacidad de acceso remoto y múltiple a través de teclados de máquinas de escribir conectados con el mecanismo de cálculo de Nueva York mediante cables telefónicos. Impresionó al público, especialmente por su forma "humana" de funcionar: ¡después de que se le formulaba una pregunta a la calculadora, parecía hacer una pausa de algunos segundos, como pensando, antes de dar la respuesta!

Muchos dispositivos menores de hardware también se originaron en Bell, como los colchones de aire flotantes que se emplean en las cabezas de cinta magnética, y los amplificadores de feedback negativo. Pero el invento más famoso fue el transistor, que crearon Bardeen, Brattain y Shockley en 1947 (véase p. 47). Fue el transistor lo que hizo posible la segunda generación de ordenadores.

PARA JUGAR A LO GRANDE (INSTANTANEAMENTE)

Presentamos **el Interface 2 ZX** Pensado y diseñado por SINCLAIR para unirse a la perfección con tu microordenador Spectrum.

Si a la hora de elegir tu microordenador optaste por el mejor, es lógico que elijas ahora el Interface 2 ZX

Ya habrás podido deleitarte con la más amplia variedad de juegos existentes para tu Spectrum (la más extensa del mercado). Ahora con el Interface 2 ZX vas a tener más ventajas para tu Spectrum:

- Podrás conectar Joysticks para sacarle, aún, mayor rendimiento a tus mejores juegos y divertirte con aquellos exclusivamente disponibles en **Cartuchos ZX:** correr, saltar, volar... a lo grande.
  ¡Menuda diferencia!
- Además, al ser cartuchos con memoria ROM, podrás, con tu SPECTRUM de 16 K, jugar con programas hasta ahora reservados para 48 K, sin ampliar la memoria.
  ¡Vaya ahorro!
- Al conectar el Interface 2 ZX tienes la certeza de poseer un periférico pensado por SINCLAIR para SINCLAIR. Tu microordenador queda a salvo de circuitos poco fiables.
  ¡Un alivio!
- Al adquirir el Interface 2 ZX y los Cartuchos ZX en la red de Concesionarios Autorizados, podrás exigir la tarjeta de garantía INVESTRONICA, única válida en territorio nacional.
  ¡Una tranquilidad!



Interface 2 ZX y Cartuchos ZX

Si aún no los tienes
no sabes lo que te pierdes

Solicita una demostración en cualquier Concesionario Autorizado INVESTRONICA.



CENTRAL COMERCIAL: Tomás Bretón, 60
Tel. 468 03 00 Telex: 23399 IYCO E Madrid.
DELEGACION CATALUÑA: Camp, 80 - Barcelona - 22

UNION PERFECTA

Así se comportan los periféricos creados por SINCLAIR para SINCLAIR: de forma perfecta. Y es lógico.

Cada vez que SINCLAIR diseña un microordenador, no lo hace de una manera aislada. Simultáneamente crea todos esos periféricos que van a hacer más potente, preciso y útil el microordenador que tiene entre manos.

Periféricos pensados y diseñados para dar un servicio óptimo, pero con un precio razonable, dentro de la filosofía SINCLAIR:

"Hacer la informática accesible a todos".

Por eso cuando creó el ZX 81 vio la necesidad de dotarlo con una ampliación de memoria de 16K RAM para que no quedara pequeño y de una impresora sencilla y barata pero útil y precisa.

Así es la filosofía SINCLAIR. Así son los periféricos de SINCLAIR para SINCLAIR.

Microordenadores
sinclair
Toda una filosofía.



CENTRAL COMERCIAL: Tomás Bretón, 60
Tel. 468 03 00 Telex: 23399 IYCO E Madrid.
DELEGACION CATALUÑA: Camp, 80 - Barcelona - 22